Sortowanie przez scalanie w języku C++

Źródło: Xavi Cabrera, domena publiczna.

Poznaliśmy już rekurencyjny algorytm sortowania przez scalanie, wykorzystujący metodę „dziel i zwyciężaj". Jest on jednym z najszybszych sposobów sortowania.

W tym e-materiale zajmiemy się implementacją algorytmu *merge sort* w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Sortowanie przez scalanie w języku Java,
- Sortowanie przez scalanie w języku Python.

Więcej zadań? Sięgnij do: Sortowanie przez scalanie – zadania maturalne.

Twoje cele

- Przeanalizujesz, czym jest rekurencja oraz metoda „dziel i zwyciężaj".
- Scharakteryzujesz działanie algorytmu sortowania przez scalanie.
- Napiszesz program wykorzystujący ten algorytm.

Film samouczek

Problem 1

Napisz program niemalejąco sortujący wyniki ankiety dotyczącej równości w życiu społecznym, zrealizowanej przez CBOS w 2000 roku. Wykorzystaj w tym celu sortowanie przez scalanie (*merge sort*).

Procentowe wyniki ankiety „Czy pana/pani zdaniem równość w społeczeństwie powinna czy też nie powinna oznaczać, że wszyscy obywatele mają równe szanse zdobycia wykształcenia i osiągnięcia wysokiej pozycji społecznej, o ile mają odpowiednie zdolności i chęci?", zrealizowanej w lutym 2000 roku przez CBOS:

- 67% odpowiedziało: „Zdecydowanie tak".
- 21% odpowiedziało: „Raczej tak".
- 2% odpowiedziało: „Trudno powiedzieć".
- 3% odpowiedziało: „Zdecydowanie nie".
- 7% odpowiedziało: „Raczej nie".

Specyfikacja:

Dane:

- `dane[ ]` – tablica liczb naturalnych

Wynik:

- Na standardowym wyjściu wyświetlane są wszystkie elementy tablicy `dane`, posortowanej w porządku niemalejącym, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Wypisz elementy tablicy posortowane niemalejąco za pomocą metody sortowania przez scalanie.

```cpp
#include <iostream>

using namespace std;

int dane[] = {67, 21, 7, 3, 2};
int liczbaElementow = 5;

void sortowaniePrzezScalanie(int indeksPoczatkowy, int indeksKoncowy) {
    // tutaj zaimplementuj sortowanie przez scalanie
}

int main()
{
```

```cpp

```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/RsjejBvogjsQu

Film nawiązujący do treści materiału: Sortowanie tablicy metodą przez scalanie.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 1.26 KB w języku polskim

Słownik

rekurencja

metoda programowania w dowolnym języku, polegająca na wywołaniu przez funkcję samej siebie; rekurencję można porównać do pętli, ponieważ jest ona wywoływana określoną liczbę razy (do momentu, gdy zajdzie warunek stopu)

sortowanie przez scalanie

podzielenie sortowanej struktury danych na dwie części, następnie na rekurencyjnym sortowaniu każdej z tych części i ponownym scaleniu już posortowanych części w jedną całość

Prezentacja multimedialna

Sortowanie przez scalanie możemy zaimplementować na wiele sposobów. W prezentacji pokażemy kod, w którym wydzielono oddzielną funkcję służącą do scalania tablicy, po zakończonym procesie podziału. Aktualnie przetwarzane dane będziemy przechowywać w tymczasowych tablicach, usuwanych z pamięci po zakończeniu działania funkcji rekurencyjnej. Sortowaną tablicę będziemy przekazywać jako jeden z parametrów wywołania funkcji.

> **Polecenie 1**
>
> Zapoznaj się z prezentacją. Przygotuj specyfikację dla przedstawionego problemu.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Implementowanie algorytmu w C++ zaczniemy od stworzenia funkcji rekurencyjnej `sortuj`, która jako argumenty przyjmuje tablicę do posortowania, indeks początku oraz indeks końca analizowanej właśnie części.

Funkcja przestanie tworzyć kolejne wywołania rekurencyjne w momencie, w którym przyjmie ona jako argument tablicę jednoelementową. W takiej sytuacji funkcja kończy działanie.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Jeżeli w badanej części będzie tylko jeden element, `indeksp` będzie równy `indeksk`.

Tak długo, jak w tablicy znajdować się będzie więcej niż jeden element, będziemy wykonywać opisane na poprzedniej stronie kroki dla wywołań funkcji rekurencyjnej.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Wspomniane kroki to: podział na dwie części, podział lewej części, podział prawej części oraz złączenie części.

Łączenie części wymaga zapisania relatywnie dużej ilości kodu, więc do operacji tej napiszemy osobną funkcję `scal`.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Zadeklarujemy więc funkcję `scal` w następujący sposób:

Oprócz tablicy, zmienne, które przyjmie funkcja to:

- `indeksp` – odpowiada pierwszemu indeksowi lewej części,

- `indeksgraniczny` – ostatni indeks lewej części,

- `indeksk` – ostatni indeks prawej części.

Wśród zmiennych brakuje pierwszego indeksu prawej części – to dlatego, że jest to indeks

o jeden większy od tego w zmiennej `indeksgraniczny` – jeśli będziemy go potrzebować, posłużymy się wartością tej właśnie zmiennej.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Aby swobodnie zamieniać elementy miejscami, stworzymy kopie lewej oraz prawej części tablicy. Dzięki kopiowaniu nie będziemy nadpisywać danych w tablicy. Aby stworzyć kopie, najpierw musimy określić rozmiar lewej oraz prawej części.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Znając już rozmiar lewej oraz prawej części, możemy zainicjować tablice, które będą przechowywać kopie elementów znajdujących się w tych właśnie częściach tablicy.

Kolejnym krokiem będzie wypełnienie tablic wartościami z lewej oraz prawej części.

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Wypełniamy nowe tablice wartościami z odpowiednich indeksów z tablicy.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Zanim przejdziemy do scalania, musimy zainicjować kilka zmiennych pomocniczych.

Pierwsze dwie zmienne będą odpowiedzialne za przechowywanie bieżącego indeksu każdej z części. Jeżeli przeniesiemy element z lewej części, zmienna `teraz_indekslewej` zostanie powiększona o jeden i będzie wskazywać na pierwszy nieprzeniesiony jeszcze element lewej części.

Zmienna `teraz_indeksorg` wskazuje miejsce w oryginalnej tablicy, gdzie zostanie umieszczony kolejny element.

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Mając już kopie tablic oraz zmienne pomocnicze, możemy przejść do scalania części. Zaczniemy od porównania elementów dwóch części. Dopóki obie części mają przynajmniej po jednym elemencie, porównujemy je i przenosimy większy.

Sprawdzimy teraz, czy pierwszy element lewej części jest większy lub równy prawemu. Jeżeli jest to pierwszy element z lewej części, zostaje

przeniesiony na miejsce `teraz_indeksorg`, w przeciwnym wypadku trafia tam pierwszy element prawej części.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Scalanie jest już gotowe:

Gdy w jednej z części skończą się elementy, wszystkie pozostałe elementy z drugiej zostają przeniesione w takiej kolejności, w jakiej się znajdują.

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Wywołajmy więc naszą funkcję i sprawdźmy jej działanie.

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PPPc0G82I

Kompletny program wygląda następująco:

Polecenie 2

Opracuj notatkę zawierającą najważniejsze informacje przedstawione w prezentacji.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który na podstawie tablicy `zbior` stworzy nową tablicę i nazwie ją `kopiaZbioru`. Program powinien skopiować tablicę, zaczynając od elementu oznaczonego indeksem a i kończąc na elemencie o innym indeksie b. Przetestuj działanie programu dla indeksu początkowego 3 oraz indeksu końcowego, który jest o 3 mniejszy niż długość tablicy.

Działanie programu przetestuj dla tablicy:

```
int zbior[16] = { 43, 6, 76, 87, 54, 654, 432, 657, 765, 543, 6
```

Nowa tablica powinna składać się ze wszystkich elementów, poza trzema pierwszymi oraz trzema ostatnimi.

Specyfikacja:

Dane:

- n – liczba naturalna
- a – liczba naturalna
- b – liczba naturalna
- `zbior[]` – n-elementowa tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy tablicy `kopiaZbioru`, utworzonej zgodnie z treścią zadania, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program tworzy kopię zbioru bez trzech pierwszych oraz trzech ostatnich elementów oraz wypisuje zawartość zbioru.

```cpp
#include <iostream>
using namespace std;

int main ()
{
    int zbior[16] = {43, 6, 76, 87, 54, 654, 432, 657, 765, 543, 654, 7657, 987, 4, 43, 65};

    // Tutaj dodaj kod.

    // Do wypisywania, użyj poniższej linijki
    // for (int i = 0; i < 10; i++) cout << kopiaZbioru[i] << " ";
}
```

```
1
```

Ćwiczenie 2

Napisz program, w którym scalisz dwie posortowane tablice w jedną (również posortowaną). Wypisz utworzoną tablicę, oddzielając jej elementy znakiem spacji.

Swój program przetestuj dla następujących tablic:

```
int tablica1[6] = { 4, 65, 432, 543, 654, 7654 }
int tablica2[7] = { 2, 23, 45, 65, 75, 432, 9211 }
```

Specyfikacja:

Dane:

- `n` – liczba naturalna
- `m` – liczba naturalna
- `tablica1` – `n`-elementowa posortowana niemalejąco tablica liczb naturalnych
- `tablica2` – `m`-elementowa posortowana niemalejąco tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy posortowanej niemalejąco tablicy, będącej wynikiem scalenia tablic wejściowych. Elementy są oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program łączy dwie tablice w jedną.

```
#include <iostream>
using namespace std;

int main ()
```

```
{
    int tablica1[6] = { 4, 65, 432, 543, 654, 7654 };
    int tablica2[7] = { 2, 23, 45, 65, 75, 432, 9211 };

    // Tutaj dodaj kod

    // Do wypisywania użyj funkcji cout
}
```

```
1
```

Ćwiczenie 3

Napisz program, w którym posortujesz malejąco daną tablicę `tablica`, używając algorytmu sortowania przez scalanie. Wypisz jej elementy, oddzielając je znakiem spacji.

Działanie programu przetestuj dla tablicy:

```
int tablica[31] = { 4, 656, 65, 43, 65, 876, 543, 756, 435, 654
```

Specyfikacja:

Dane:

- n – liczba naturalna
- `zbior[]` – n-elementowa tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy posortowanej malejąco tablicy `tablica`, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program sortuje tablicę, używając algorytmu sortowania przez scalanie.

```
#include <iostream>
using namespace std;

int main ()
{
        int tablica[31] = { 4, 656, 65, 43, 65, 876, 543, 756, 435, 654, 534, 234, 756, 432, 765, 543, 987, 423, 123, 423, 654, 756, 534, 6755, 534, 4756, 65, 4, 654, 54, 75 };

        // Tutaj dodaj kod.

```

```
10          // Do wypisywania, użyj poniższej linijki
11          // for (int i = 0; i < 31; i++) cout << tablica[i];
```

```
1
```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie przez scalanie w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

e) sortowania ciągu liczb przez scalanie,

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz, czym jest rekurencja oraz metoda „dziel i zwyciężaj".
- Scharakteryzujesz działanie algorytmu sortowania przez scalanie.
- Napiszesz program wykorzystujący ten algorytm.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Sortowanie przez scalanie w języku C++". Uczniowie mają zapoznać się z treściami w sekcji „Film samouczek" i wykonać obliczenia na podstawie dołączonych danych.

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie analizują film z sekcji „Film samouczek" wyświetlony na tablicy. Wybrany uczeń czyta treść polecenia nr 1: „Napisz program sortujący wyniki ankiety dotyczącej równości w życiu społecznym, zrealizowanej przez CBOS w 2000 roku. Wykorzystaj w tym celu sortowanie przez scalanie (merge sort)" i omawia przykładowe rozwiązanie postawionego problemu.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie zapoznają się z jego treścią. Zapisują ewentualne problemy i pytania. Po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1: „Napisz program, w którym utworzysz kopię tablicy `zbior` bez pierwszych oraz ostatnich trzech elementów. Nazwij ją `kopiaZbioru`. Następnie wypisz jej elementy, oddzielając je znakiem spacji", a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2: „Napisz program, w którym scalisz dwie posortowane tablice w jedną (również posortowaną). Wypisz utworzoną tablicę oddzielając jej elementy znakiem spacji" z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku C++.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 3: „Napisz program, w którym posortujesz malejąco daną tablicę `tablica`, używając algorytmu sortowania przez scalanie. Wypisz jej elementy, oddzielając je znakiem spacji” z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Treści w sekcji „Film samouczek” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.